Generating Compact Classifier Systems Using a Simple Artificial Immune System

Kevin Leung, France Cheong, and Christopher Cheong

***Abstract*—Current artificial immune system (AIS) classifiers have two major problems: 1) their populations of B-cells can grow to huge proportions, and 2) optimizing one B-cell (part of the classifier) at a time does not necessarily guarantee that the B-cell pool (the whole classifier) will be optimized. In this paper, the design of a new AIS algorithm and classifier system called simple AIS is described. It is different from traditional AIS classifiers in that it takes only one B-cell, instead of a B-cell pool, to represent the classifier. This approach ensures global optimization of the whole system, and in addition, no population control mechanism is needed. The classifier was tested on seven benchmark data sets using different classification techniques and was found to be very competitive when compared to other classifiers.**

***Index Terms*—Artificial immune systems (AISs), classification, instance-based learning (IBL).**

I. Introduction

CLASSIFICATION is a commonly encountered real-world decision-making task. Categorizing an object into a predefined group or class based on a number of observed attributes of the object is a typical classification problem [1]. While there have been many studies [2], [3] on statistical classifiers such as discriminant analysis (DA) and logistic regression, researchers are now focusing more on artificial intelligence (AI) techniques such as genetic algorithms and artificial neural networks (ANNs) as classifier systems [4], [5].

A more recent form of AI technique known as artificial immune system (AIS) is rapidly emerging. It is based on natural immune system principles, and it can offer strong and robust information processing capabilities for solving complex problems. Just like ANNs, AIS can learn new information, recall previously learned information, and perform pattern recognition in a highly decentralized manner [6]. There are various applications of AIS, and they include data analysis [7], [8], scheduling [9], machine learning [10], [11], classification [12], fault detection [13], and security of information systems [14].

However, there are two fundamental problems with current AIS classifier systems. The first problem is related to population control, where it has been found that the number of B-cells, which match some antigens, increases, through cloning and mutation, to such an amount that it usually overtakes the entire population of B-cells [15]. In addition, most AIS classifier systems make use of populations of B-cell pools, and the problem that was identified is that optimizing one B-cell (which is only part of the classifier) at a time does not necessarily guarantee that the B-cell pool (which is the complete classifier) will be improved.

This paper introduces a new AIS algorithm and classifier system that resolves the two problems mentioned previously without sacrificing the classification performance of the system. The rest of this paper is organized as follows: Section II provides some background information on how the immune system works. Section III discusses some related work, while Section IV provides an explanation of the algorithm and implementation of the new classifier system. Section V provides details of the tests performed and the results obtained, and Section VI concludes this paper.

II. Background Information

Before discussing some related AIS works and explaining the design and implementation of the new AIS algorithm, background information on the main immune system metaphors is provided to aid in understanding the concepts of AIS classifiers.

A. Natural Immune System

The immune system protects the body from foreign substances called antigens by recognizing and eliminating them by a process that is known as the immune response. It makes use of a huge variety of antibodies to neutralize these antigens [16]. These antibodies are proteins that are produced by B-cells. Each B-cell produces a single type of antibody and is thus specific to that particular antigen.

The B-cells form what is known as the immune network. This network acts to ensure that, once useful B-cells are generated, they remain in the immune system until they are no longer required. When a B-cell encounters an antigen, an immune response is elicited, and the antibody will try to bind itself with the antigen, so that the latter one can be neutralized. If the affinity between the antibody and the antigen is sufficiently high, its B-cell becomes stimulated, resulting in the production of mutated clones. As these new B-cells are added to the immune network, an equal quantity of the least stimulated B-cells is removed or dies. This is, in fact, based on the Darwin principle (survival of the fittest), and it thus maintains diversity in the immune system [17].

B. Primary and Secondary Response

The immune system has two types of response: 1) primary and 2) secondary. The primary response occurs when the immune system encounters an antigen for the first time and reacts against it by producing antibodies. The immune system learns

Manuscript received December 20, 2006; revised March 20, 2007. This paper was recommended by Associate Editor H. Takagi.

The authors are with the School of Business Information Technology, RMIT University, Melbourne Vic. 3000, Australia (e-mail: kevin.leung@rmit.edu.au; france.cheong@rmit.edu.au; chris.cheong@rmit.edu.au).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCB.2007.903194

about the antigen and thus prepares the body against any further invasion of that antigen. This learning mechanism creates what is called the immunological memory.

The secondary response occurs when the immune system encounters an antigen against which it has reacted before. It is characterized by a shorter lag phase, a higher rate of antibody production, and longer persistence of antibody synthesis since the immune system already has all the information on the antigen from the immunological memory.

There are two ways by which the memory is achieved in the immune system. The most widely held view uses the concept of "virgin" B-cells being stimulated by antigen and producing memory cells and effector cells. A theory that is less accepted by experimental immunologists but held by some theoretical immunologists uses the concept of immune network theory, which is the approach that will be adopted in this paper.

C. Immune Network Theory

The immune network theory was first proposed by Jerne [18] and reviewed by Perelson [19]. This theory proposes that the immune system maintains an idiotypic network on interconnected B-cells for antigen recognition. The strength of the B-cells connections is directly proportional to the affinity that they share, and the cells can both stimulate and suppress each other in order to stabilize the network. Due to the fact that the immune system can only have a limited number of cells available to it at any one time, it must replace a percentage of its cells on a continuing basis. This replacing or self-organizing process ensures the stability of the network. The way that the B-cells stimulate and cancel each other is explained in the following sections.

D. Clonal Selection Theory

The clonal selection theory describes the basic features of an immune response to an antigen stimulus. It involves two processes: 1) pattern recognition and 2) selection [20]. It establishes the idea that only those cells that can recognize an antigen are able to gain in concentration and affinity, while those that do not die out. The theory makes use of the concept of affinity maturation.

E. Affinity Maturation

When an antibody on the surface of a B-cell binds to an antigen, the B-cell becomes stimulated. The level of stimulation depends on not only how well it matches the antigen but also how it matches other B-cells in the immune network. If the stimulation level rises above a given threshold, the B-cell will start replicating itself, producing clones of itself. An important aspect of this cloning process is that it does not produce exact clones. The offsprings that are produced by this cloning process are mutated. The newly mutated cells may have a better match for the antigen and will thus proliferate and survive longer than existing B-cells. By repeating the processes of mutation and selection, the immune system learns to produce better matches for the antigen. Alternatively, if the stimulation level is below a given threshold, the B-cell will not replicate, and in time, it will die off. This whole process of mutation and selection is called affinity maturation [21].

III. Related Work

This section gives a description of the main works that are related to artificial immune classifiers.

A. Clonal Selection Algorithm (CSA/CLONALG)

The CSA was first developed by de Castro and Von Zuben [22], and it uses the concept of clonal selection theory. It was later improved and renamed as CLONALG [23].

CLONALG works by retaining only one memory cell for each antigen that is presented to it and makes use of a ranking system to determine the rate at which clones can be produced. The clones, in turn, are mutated using a multipoint mutation method, whereby they are mutated if a randomly generated control number exceeds a given threshold. De Castro and Von Zuben [23] suggested that CLONALG could be used for pattern recognition and experiment with it by using binary data. White and Garrett [24] made some modifications to CLONALG and renamed it to CLONCLAS, and found out that it could successfully classify previously unknown patterns.

B. Resource Limited AIS (RLAIS)/Artificial Immune Network (AINE)

One of the first AIS created by Timmis *et al.* [7] was an effective data analysis tool. It achieved good results by classifying benchmark data into specific clusters. However, a number of problems were clear: 1) the population control mechanism was not efficient in preventing an exponential population explosion with respect to the network size, and 2) the resultant networks became so large that it was difficult to interpret the data and identify the clusters [25]. To address the issues that are raised by their first AIS, the authors proposed a new system called the RLAIS, which was later renamed to AINE.

RLAIS uses artificial recognition balls (ARBs), which was inspired by the work of Farmer *et al.* [26] in describing antigenic interaction within an immune network. The ARBs represent a number of identical B-cells, and a link is created between two B-cells if the affinity (distance) between two ARBs is below a certain network affinity threshold. The ARBs must compete for these B-cells based on their stimulation level, and those that are left with no B-cells are considered weak and consequently removed from the network. Thus, they reduce complexity and redundant information in the network [27]. However, even though ARBs are essentially a compression mechanism that takes B-cells to a higher granularity level, the population of B-cells still grows rapidly to huge proportions [15]. A limited number of B-cells were predefined in RLAIS in order to solve this problem and to effectively control the population expansion.

C. Self-Stabilizing AIS (SSAIS)

SSAIS was developed in an attempt to solve the problems of RLAIS. Neal [8] identified two fundamental problems as: 1) the nature of the resource allocation mechanism and 2) the noncontinuous nature of the synchronous update mechanism. The second RLAIS problem is that it does not lend itself to a genuinely continuous mode of operation as the resource

allocation mechanism was performed for each pass through the data set. This problem is of minor importance to this appear, and it was easily resolved by reviewing the RLAIS algorithm and making sure that all the functionalities of the algorithm were fully operational after the presentation of every data item.

The first problem relates to the ARBs. Unlike the natural immune system, which takes time to build up immunology and lose it again, an ARB in RLAIS could gain or lose all of its resources in one pass through the network. In addition, the nature of the resource allocation mechanism requires the normalization of the level of stimulation, leading to unnecessary complex calculations. The first problem was resolved simply by making changes to the way the ARBs work. In RLAIS, a predefined number of resources was allocated to ARBs, in proportion to their stimulation level. On the other hand, in SSAIS, there is no fixed number of resources that are distributed among the ARBs. In fact, each ARB can increase its own resource allocation by registering the highest stimulation level for any incoming data item and increment the resources it holds by adding its current stimulation level [8]. Since there is no longer a limited number of resources in SSAIS, it makes use of a mortality constant and a decay rate to control the network population size.

D. Artificial Immune Recognition System (AIRS)

The main study, which regards AIS as a supervised classifier system, was done by Watkins [12]. The classifier system was named AIRS, and it was based on the principle of RLAIS and made use of ARBs. AIRS is a very powerful classification tool, and when compared to the 10–30 best classifiers on publicly available classification problem sets, it was found to be among the top five to eight classifiers for every problem set, except one in which it ranked second [28]. As such, AIRS has proved to be a successful general-purpose classifier.

The AIRS algorithm is shown in Algorithm 1. AIRS is considered to be a one-shot learning algorithm, in that the training data (antigens) are presented to the system only once [28]. It uses a pool of ARBs, some of which are mutations of an existing B-cell and some of which are simply randomly generated cells. The stimulation level is inversely proportional to the distance in the feature space, i.e., the smaller the Euclidean distance between an antigen and an ARB, the greater the stimulation. The ARBs, which are most highly stimulated by exposure to the antigen, are cloned and mutated. Once all the ARB matching has been done, it is usually the case that the limit for available resources is exceeded, in which case the ARBs with the lowest affinity are removed from the ARB pool. This process continues over several generations, with the most stimulated cells being retained and the least stimulated ones being eliminated.

Once the average stimulation level of the entire population reaches a threshold level, the process stops, and the best ARBs from the same classification class to which the antigen belongs are compared. If that ARB has a higher affinity than the best matching memory cell, then it becomes a candidate for promotion to a memory cell and is added to the pool of memory cells. In addition, if the new candidate cell is sufficiently similar to the memory cell that originally was the most stimulated by the invading antigen, the old memory cell is replaced in favor of the new and more accurate memory cell. This mechanism contributes both to the generalization capabilities of the AIRS classifier and to the data reduction capabilities of the system [28]. In a typical AIRS training process, there are fewer than half as many memory cells in the AIRS system as there are training instances, and only a small fraction of the memory cell population is identical to the training instances [29].

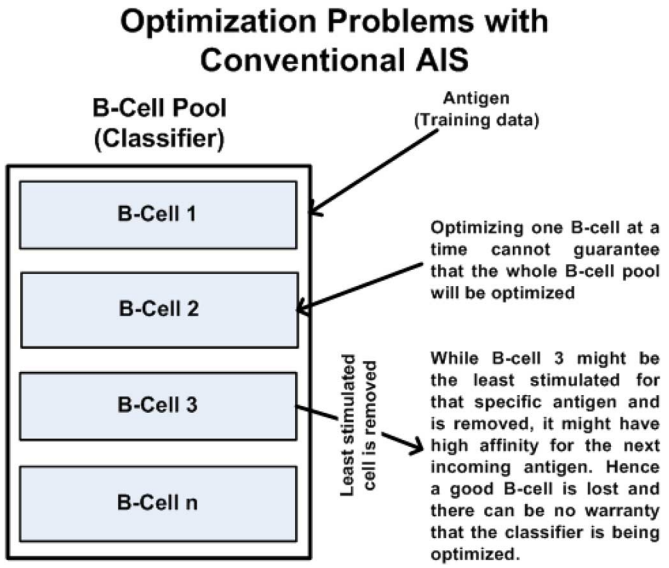

Fig. 1. Optimization problems found in conventional AIS.

Algorithm 1: AIRS Algorithm
Load antigen population {training data}
for each antigen in population **do**
Select the memory cell that has the highest affinity to the antigen from the memory cell pool
Create a pool of B-cells, which consists of the offsprings of the selected memory cell
repeat
Clone and mutate most highly stimulated B-cell
Remove least stimulated B-cells
until stimulation level of B-cell pool $>$ threshold
Compare affinity between the best B-cell and antigen against affinity between the memory cell and antigen
if affinity of the best B-cell $>$ affinity of best memory cell
then
Add the best B-cell to the memory pool
Remove the memory cell from the memory pool
end if
end for

One major problem that was identified with the AIRS learning algorithm is that it does not guarantee the generation of an optimal classifier. This is so because optimizing one B-cell at a time cannot assure that the B-cell pool, which is the complete classifier, will be optimized as well. In addition, AIRS controls its population size by removing the least stimulated cells from its B-cell pools; however, these cells, being the least stimulated for one specific antigen, might have very high affinities with other antigens, leading to a loss of good B-cells. Thus, optimizing one part (B-cell) of the system might have a negative effect on the immune system as a whole. The two main optimization issues identified are summarized in Fig. 1.

E. Other AIS Classifiers

There are many other AIS classifiers that have been used in a variety of different settings. The *Jisys* system, for instance, was used in mortgage fraud detection by classifying applicants into either fraudulent or nonfraudulent groups [30]. Moreover, while Zheng *et al.* [31] developed an AIS to detect and discriminate texture objects from satellite images, Zhong *et al.* [32] proposed a new unsupervised artificial immune classifier to perform remote sensing image classification. Another AIS classifier model, which is known as Self-Organizing Sparse Distributed Memories and was developed by Hart and Ross [33], was used for clustering nonstationary data.

IV. Simple AIS (SAIS)

This section presents an overview of the proposed algorithm and classifier system named SAIS. As the name implies, SAIS is very simple in that it adopts only the concept of affinity maturation, which deals with stimulation, cloning, and mutation as opposed to the currently available AIS, which tends to focus on several particular subsets of the features that are found in the natural immune system. It also generates a compact classifier using only a predefined number of exemplars per class. This will be discussed further in the next section.

A. Definitions

This section defines the key terms and concepts used as they are applied to the SAIS algorithm. Some of these definitions are taken from the paper by Watkins *et al.* [28].

- *B-cell*: In this paper, it is referred to as an exemplar.
- *Class*: Classification category of an instance of data.
- *clonalRate*: An integer value that is used to determine the number of mutated clones that an exemplar is allowed to produce.
- *Exemplar*: Also known as B-cell, it represents the whole classifier.
- *hyperMutationRate*: An integer value that is used to determine the number of mutated clones that are generated into the cell population.
- *numClasses*: Number of classes (depends on the data set).
- *numClones*: Number of clones that can be created (*numClones* = *clonalRate* × *hyperMutationRate*).
- *numExemplars*: Number of exemplars. In this paper, only one exemplar is used.
- *numAttb*: Number of attributes (depends on the data set).
- *maxIteration*: Maximum number of iterations (applies to the training process).
- *numCorrect*: Number of correctly classified data.
- *percentCorrect*: Percentage of correctly classified data.
- *probMutation*: Probability that a given clone will mutate.
- *Testing data*: Data set used to test the SAIS classifier.
- *Training data*: Data set used to train the SAIS classifier.

B. Algorithm

This section describes the differences between a conventional algorithm and the SAIS algorithm. It also provides the pseudocode that explains how the SAIS model works.

1) Conventional AIS Algorithm: In a conventional AIS algorithm, a classifier system is constructed as a set of B-cells that can be used to classify a wide range of data, and in the context of immunology, the data to be classified is known as antigens. A typical AIS algorithm operates in three steps.

1) First, a set of training data (antigens) is loaded, and an initial classifier system is created as a pool of B-cells with attributes that were initialized from either random values or values taken from random samples of antigens.
2) Next, for each antigen in the training set, the B-cells in the cell pool are stimulated. The most highly stimulated B-cell is cloned and mutated, and the best mutant is inserted in the cell pool. To prevent the cell pool from growing to huge proportions, B-cells that are similar to each other and those with the least stimulation levels are removed from the cell pool.
3) The final B-cell pool represents the classifier.

The conventional AIS algorithm is shown in Algorithm 2. From the description of the algorithm, three problems are apparent with conventional AIS algorithms.

1) Only one pass through the training data does not guarantee the generation of an optimal classifier.
2) Finding optimal B-cells does not guarantee the generation of an optimal classifier, as local optimizations at the B-cell level does not necessarily imply global optimization at the B-cell pool level (see Fig. 1).
3) The simple population control mechanism of removing duplicates cannot guarantee a compact B-cell pool size. Many of the early AIS classifiers that are reported in the literature [15], [27] suffer from the problem of huge size. Good B-cells may be lost during the removal process (see Fig. 1). A conventional AIS classifier was experimented with, and the size of the cell pool was found to grow to astronomical proportions when using such a simple population control mechanism.

Algorithm 2: Conventional AIS Algorithm—Main

```
Load antigen population {training data}
Generate pool of B-cells with random values or values from random antigens
for each antigen in population do
    Present antigen to the B-cell pool
    Calculate stimulation level of B-cells
    Select most highly stimulated B-cell
    if stimulation level > threshold then
        Clone and mutate selected B-cell
        Select best mutants and insert into the B-cell pool
    end if
    Delete similar and least stimulated B-cells from the B-cell pool
end for
Classifier ← B-cell pool
```

2) SAIS Algorithm: In order to address the issues that are present in conventional AIS algorithms, the SAIS algorithm is designed to operate in four steps.

1) First, a set of training data (antigens) is loaded, and an initial classifier system is created as a single B-cell containing a predefined number of exemplars that are initialized from random values. The purpose and content

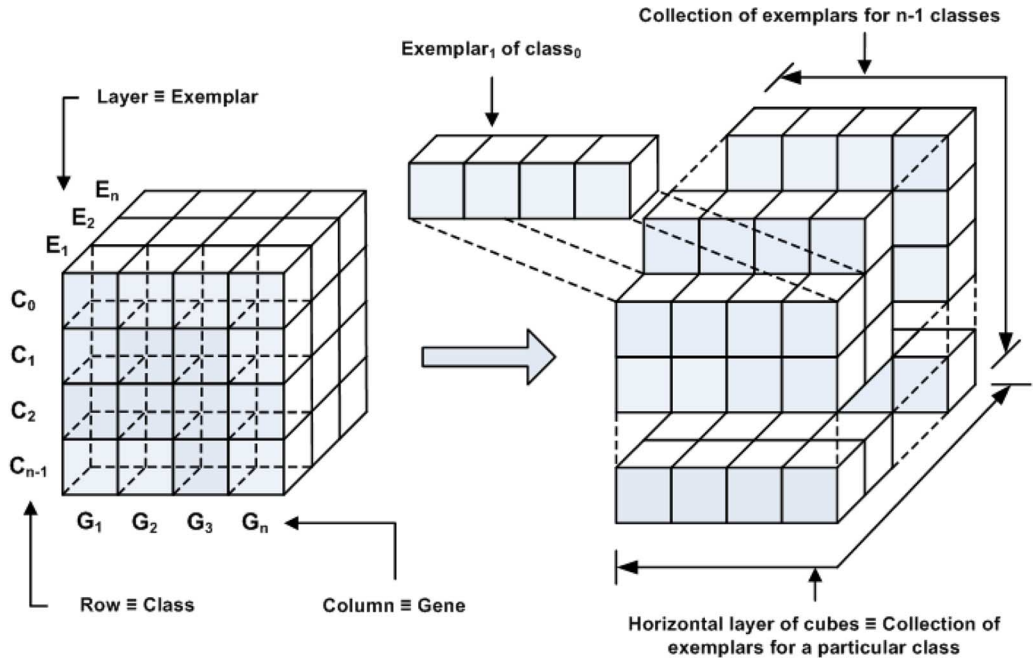

Fig. 2. Structure of the 3-D array used in the evolution process of SAIS.

of this B-cell is different from the one used in conventional AIS algorithms. This B-cell represents the complete classifier, and it contains one or more exemplars per class to classify data. A B-cell in a conventional AIS algorithm, however, represents exactly one exemplar, and the complete classifier is made up of a pool of B-cells.

2) Next, an evolution process is performed and iterated until the best possible classifier is obtained. The current B-cell is cloned, and the number of clones that can be produced is determined by the clonal rate and hypermutation rate. Mutants are then generated by using the hypermutation process that is found in natural immune systems. More specifically, this is achieved by randomly mutating the attributes of each clone that was created and storing them in 3-D array. Such an array is used because it is easier to store the attributes, classes, and exemplars. Fig. 2 shows the structure of the 3-D array, where each row represents a class, each column represents an attribute, and each layer represents an exemplar. It should be noted that the 3-D array will collapse to a 2-D array when a single exemplar is used since there will be only one layer of cells. The evolution process of SAIS is shown in Algorithm 3.

Algorithm 3: SAIS Algorithm—Population Evolution

```
numClones ← clonalRate × hyperMutationRate
for all i such that i < numClones do
  for all exemplars do
    for all classes do
      for all attributes do
        if random number generated > probMutation
        then
          Get random number
          Calculate mutatedValue of attribute using the random
          number
        end if
        Set mutatedValue to clone
      end for
    end for
  end for
end for
```

3) Each mutant is then evaluated by using the classification performance. The classification performance is a measure of the percentage of correctly classified data. If the classification performance of the best mutant is better than that of the current B-cell, then the best mutant is taken as the current B-cell. The measure of stimulation is different from the one used in conventional systems, in that a classification performance is used as a measure of stimulation of the complete classifier on all the training data rather than the distance (or affinity) between part of the classifier (a B-cell) and part of the data (an antigen). The evaluation process of SAIS has a linear computational complexity and is shown in Algorithm 4.

Algorithm 4: SAIS Algorithm—Performance Evaluation

```
for all all mutants do
  Get actual classification of data set
  Classify data set and get predicted classification
  if actual classification = probMutation then
    Increment numCorrect
  end if
  percentCorrect ← numcorrect × 100 / size of dataset
  if percentCorrect of mutant > percentCorrect of current B-cell
  then
    Current B-cell ← mutant
  end if
end for
```

4) The current B-cell represents the classifier.

The SAIS algorithm is shown in Algorithm 5. The training data are first inputted in the model, and a B-cell is randomly

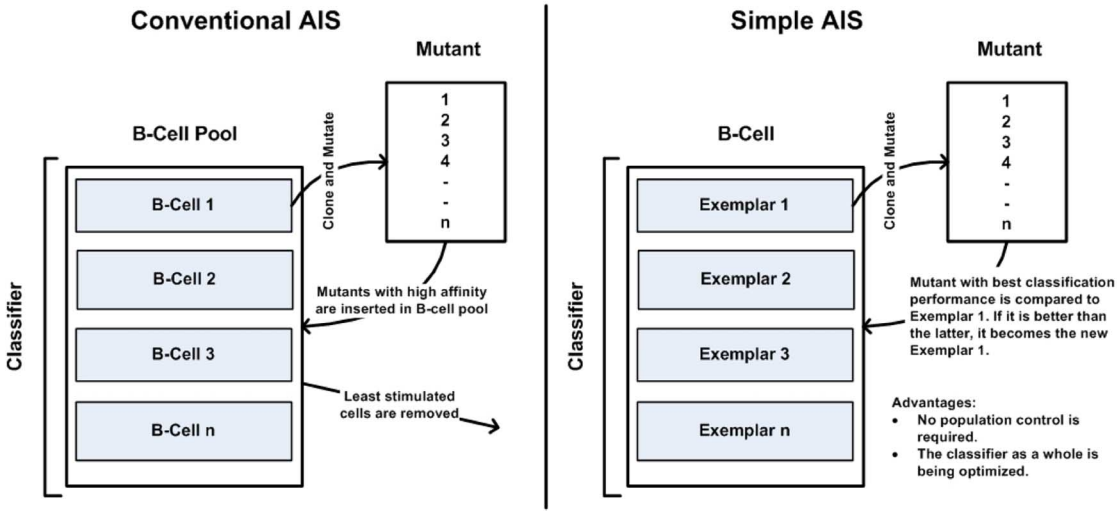

Fig. 3. Comparison of conventional AIS and SAIS.

initialized. This B-cell represents the classifier, and it is evolved through cloning and mutation, and evaluated based on its classification performance. A 100% classification performance would be ideal; however, in reality, this is not possible, and what usually happens is that the performance reaches a near-constant value, which is less than 100%. This is basically the stopping mechanism of the classifier. It was also found that an average number of 237 iterations was sufficient for the performance to reach a near-constant value; however, to be on the safe side, the *maxIteration*, which is one of the system parameters of SAIS classifier, was set to 600. These will be discussed further in Sections IV-D and V-A.

The computational complexity of an algorithm is the cost, which is measured in running time, or storage, or any other units that are relevant, of using the algorithm to solve a particular problem [34]. Based on Algorithms 3, 4, and 5, it can be shown that the evolution, evaluation, and main SAIS processes have all linear computational complexity. However, in reality, there are many factors such as the number of exemplars, classes, and attributes, which impact on the execution time of the algorithm. To be able to determine the computation complexity of SAIS, its CPU time needs to be investigated and compared to those of other classifiers. This is to be done on the same data set, under the same conditions and using the same computer. However, since the other classifiers could not be obtained, only the CPU time of SAIS could be generated.

Using a B-cell to represent the whole classifier rather than part of the classifier has several advantages.

1) Optimizations are performed globally rather than locally, and nothing gets lost in the evolution process.
2) There is no need for any population control mechanism, as the classifier consists of a small predefined number of exemplars. So far, in the experiments performed, only one exemplar per class to be classified was used. This ensures the generation of the most compact classifier possible.

Algorithm 5: SAIS Algorithm—Main
Load antigen population {training data}
Current B-cell ← randomly initialized B-cell
repeat
Evolve the B-cell by cloning and mutation {call Algorithm 3}
Evaluate mutated B-cells by calculating their classification performance {call Algorithm 4}
New B-cell ← mutated B-cell with best performance
if performance of new B-cell > current B-cell **then**
Current B-cell ← new B-cell
end if
until *maxIteration*
Classifier ← current B-cell

As a summary of this section, a diagram showing the differences between a conventional AIS and our SAIS is provided in Fig. 3.

C. Model Implementation

SAIS was implemented in Java using the Repast[1] agent-based modeling framework. Three different types of classification methods were used, and each method has a linear computational complexity. One of the methods is exemplar based, while the two others are function based. In the exemplar-based method, the attributes of a single exemplar per class are stored in the classifier. If there are two classes, for instance, the complete classifier will consist of two exemplars and their attributes. However, in the function-based method, the whole classifier consists of only one set of parameters for the function in question whether there are one or multiple classes.

1) Minimum-Distance Classification Method: In this exemplar-based method, a distance measure is used to classify the data. This approach is adapted from instance-based learning [35], which is a learning paradigm in which algorithms store the training data and use a distance function to classify the data to be tested. The heterogeneous Euclidean-overlap metric (HEOM) [36] is used. It can handle both categorical and continuous attributes and is defined as

$$totalDist(x_1, x_2) = \sqrt{\sum_{i=1}^{n} dist(x_{1,i}, x_{2,i})^2} \qquad (1)$$

[1]Available from http://repast.sourceforge.net.

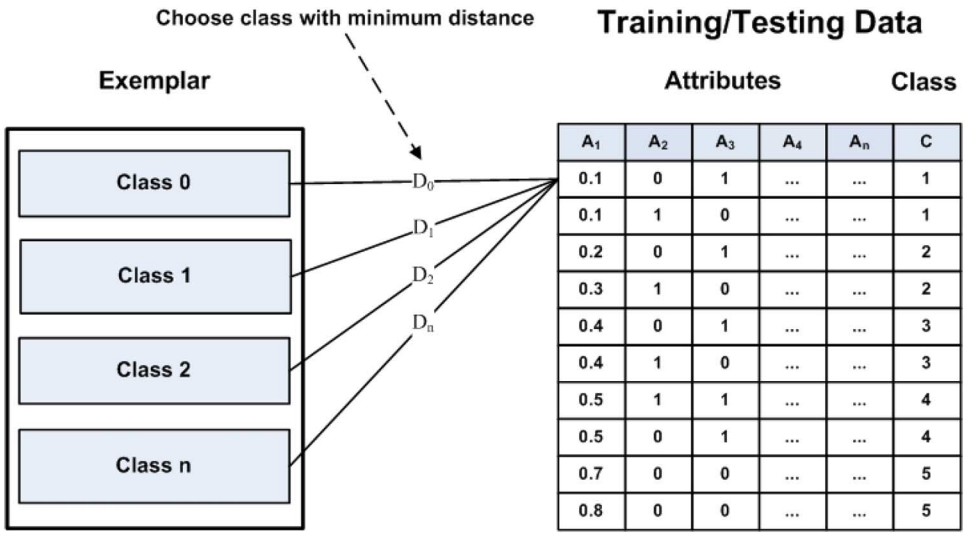

Fig. 4. Minimum-distance classification method of SAIS.

where x_1 is an exemplar, x_2 is an antigen, and n is the number of attributes. The distance between an exemplar and one antigen is calculated as

$$dist(x_{1,i}, x_{2,i}) = \begin{cases} catDist(x_{1,i}, x_{2,i}), & \text{if categorical} \\ contDist(x_{1,i}, x_{2,i}), & \text{if continuous.} \end{cases} \quad (2)$$

Categorical attributes are handled by the overlap function

$$catDist(x_{1,i}, x_{2,i}) = \begin{cases} 0, & \text{if } x_{1,i} = x_{2,i} \\ 1, & \text{otherwise} \end{cases} \quad (3)$$

while continuous attributes are handled by the Euclidean function, which is calculated as

$$contDist(x_{1,i}, x_{2,i}) = (x_{1,i} - x_{2,i}). \quad (4)$$

The minimum distance is then chosen to determine the class to classify each antigen (see Fig. 4). For example, suppose that the exemplar contains only two classes (0 and 1). SAIS will determine the distance for each class, and if the distance for class 0 is smaller than that for class 1, then SAIS will classify the data as class 0. The B-cell undergoes cloning and mutation, and the process described previously is repeated until an optimal classifier is obtained. The predicted classifications are then checked against the testing data, and the percentage of correctly classified data can thus be generated. It should also be noted that all the training data are normalized before the training process starts. This avoids the problem of overpowering the other attributes if one of them has a relatively large range.

2) DA Classification Method: A second classification method that was used is an equivalent of the DA statistical technique. DA is usually used to classify observations into two or more mutually exclusive groups by using the information provided by a set of predictor attributes. It was first proposed by Fisher [37] as a classification tool and has been reported as the most commonly used data mining technique for classification problems, despite the fact that it cannot handle independent categorical attributes properly and that it depends on a relatively equal distribution of group membership. It takes the form

$$y = c + \sum_{i=1}^{n} w_i x_i. \quad (5)$$

When using a conventional DA, each independent attribute x is multiplied by its corresponding weight w, and these products and the constant c are added together, resulting in a single discriminant score y.

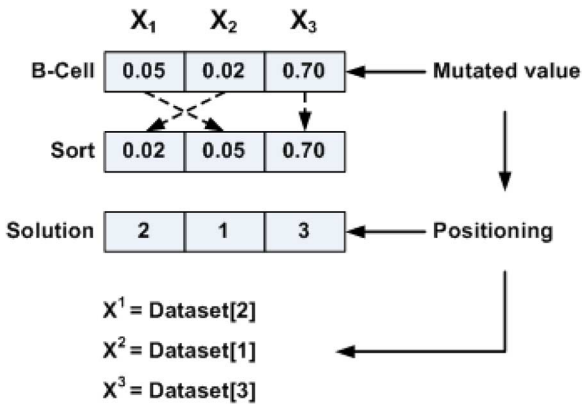

Fig. 5. Sorting of attributes for use in polynomial function.

TABLE I
SYSTEM PARAMETERS OF SAIS

Name	Description and value
clonalRate	Default value = 10.
hyperMutationRate	Default value = 10. Number of clones that can be mutated = 100 (10 × 10).
maxIterations	600 iterations were enough for the performance of the classifier to become constant. In fact, an average of 237 iterations were enough for the performance of SAIS to become constant as shown in Table III.
probMutation	Probability of mutation = 0.7.

TABLE II
DATA SETS USED FOR THE EXPERIMENTS

Dataset	Number of attributes	Type	Classes	Instances (n)	n with missing attributes	n used for experiment
Cancer	9	9 Categorical	2	699	16	683
Credit	15	6 Continuous 9 Categorical	2	690	37	653
Diabetes	8	8 Continuous	2	768	-	768
Hepatitis	19	6 Continuous 13 Categorical	2	155	75	80
Ionosphere	34	34 Continuous	2	351	-	351
Iris	4	4 Continuous	3	150	-	150
Wine	13	13 Continuous	3	178	-	178

The proposed method is identical to a conventional DA, except that the constant and the weights used in the equation are determined using AIS rather than statistical techniques. The classifier optimizes the classification score y by evolving (cloning and mutating) the constant c and the weights w of the attributes x.

3) Polynomial Classification Method: The third method used for classification purposes is a polynomial function, which is a nonlinear statistical technique. A polynomial is a mathematical expression involving the sum of a constant and powers in one or more attributes multiplied by the corresponding coefficients, i.e.,

$$y = c + \sum_{i=1}^{n} w_i x_i^i. \tag{6}$$

The polynomial technique that is used in SAIS is function based, as represented by (6). Just like the DA technique, this function is optimized by evolving the constant c and the weights w of all the different attributes x.

However, a problem that is identified in this classification method is that, while the actual output does not depend on the position of the attributes, the equation depends heavily on the order in which the attributes are entered in (6). For instance, the first and second attributes that are entered in the equation will have powers of 1 and 2 attributed to them, respectively.

To resolve this issue, all the different attributes are sorted in and by order of their corresponding mutated values. The net result is the positioning of the different attributes, as shown in Fig. 5, which also gives an example of how the sorting function works. This positioning is then used to determine how the attributes will be inserted into the SAIS model.

D. System Parameters

The system parameters used by the SAIS classifier are shown in Table I.

V. EXPERIMENTATION AND RESULT

The classification performance of SAIS was tested on seven benchmark data sets that are publicly available from the machine learning repository of the University of California, Irvine [38]. These data sets are given as follows:

1) Wisconsin Diagnostic Breast Cancer;
2) Credit Approval;
3) Pima Diabetes;
4) Hepatitis Domain;
5) Ionosphere;
6) Iris Plants;
7) Wine Recognition.

Two different sets of experiments were performed, and these are discussed in the next sections.

TABLE III
AVERAGE NUMBER OF ITERATION FOR EACH DATA SET

Dataset	Iterations
Cancer	165
Credit	234
Diabetes	335
Hepatitis	216
Ionosphere	287
Iris	258
Wine	167
Average	**237**

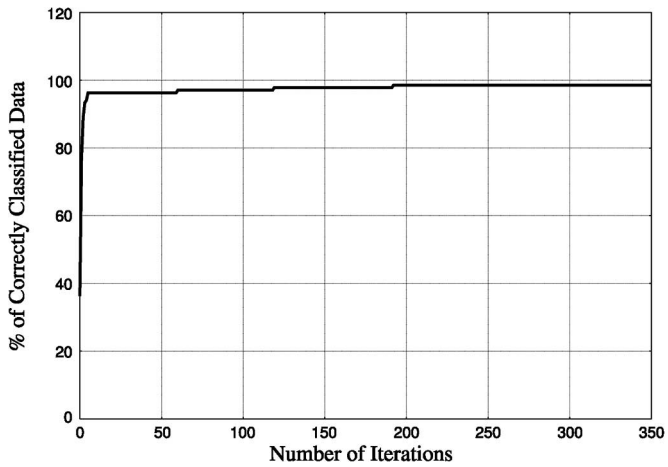

Fig. 6. Convergence speed of SAIS.

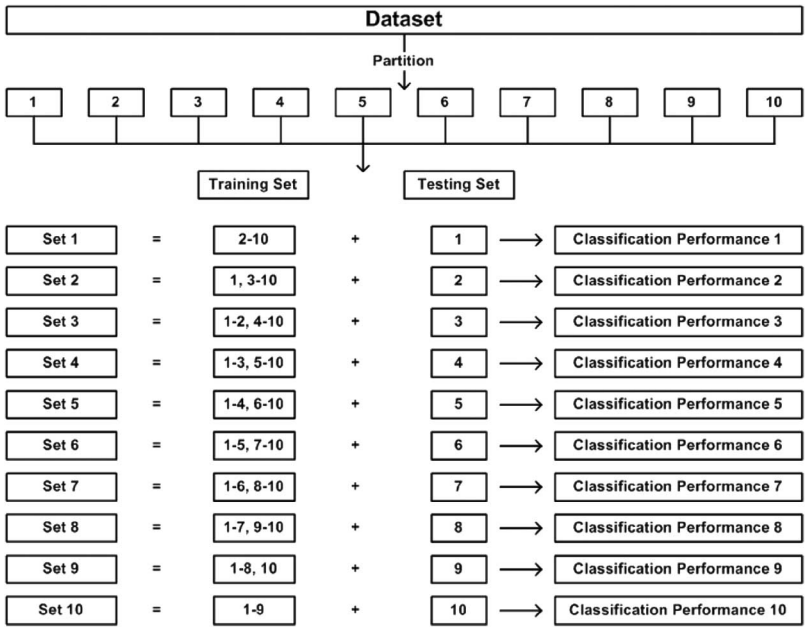

Fig. 7. Partitioning of data set (tenfold cross validation).

TABLE IV
COMPARISON OF PERFORMANCE OF SAIS USING DIFFERENT CLASSIFICATION TECHNIQUES (EXCLUDING MISSING ATTRIBUTES)

Classifier	Cancer	Credit	Diabetes	Hepatitis	Ionosphere	Iris	Wine
Minimum Distance	94.4% (2.5)	86.2% (2.6)	77.4% (5.7)	87.5% (10.2)	87.5% (4.4)	97.3% (4.7)	97.1% (4.7)
Discriminant Analyis	96.6% (2.7)	85.3% (4.1)	75.5% (6.3)	83.8% (14.4)	82.9% (5.0)	50.7% (12.2)	55.5% (10.5)
Polynomial	96.0% (3.8)	68.2% (4.2)	75.4% (5.7)	86.3% (15.4)	73.8% (5.0)	51.3% (14.8)	33.1% (10.9)

( ) - Standard deviation

A. *Experiment 1—Excluding Missing Attributes*

Three of the data sets (Breast Cancer, Credit Approval, and Hepatitis Domain data sets) contain observations with missing attributes. There are several methods of dealing with missing attributes [39], and the simplest method, which is to disregard all instances that have at least one missing attribute, was adopted. Table II shows the description of the data sets used in the first experiment.

To remain comparable with other classifiers that are reported in the literature [40], a tenfold cross-validation technique was used to partition each data set into training and testing sets. SAIS was run 600 times on the ten training sets of each data set, and results show that, on average, the performance of the classifier becomes constant after 237 iterations. Table III shows the average number of iteration of each data set that is needed for the performance of the classifier to become constant, while Fig. 6 gives an example of the convergence speed of SAIS.

The classifier was then run on the ten testing sets of each data set. Fig. 7 shows how the data sets were partitioned and how the classification performances were obtained. As shown in the figure, each data set was partitioned into ten portions, thereby generating ten different sets of data, each containing one portion as the testing set and nine portions as the training set. Data set 1, for instance, consists of portion number 1 as the testing set and portion numbers 2–10 as the training set. Each set of data produced a classification performance, and the ten classification results were averaged to yield an overall accuracy of correctly classified data. While Table IV gives the classification performance of SAIS as well as the standard deviations, Table V shows an example of the confusion matrix obtained for each data set when using the three different classification techniques.

Based on the experiments that were performed, it was found that the minimum-distance method is best for data sets with continuous attributes (e.g., Diabetes, Ionosphere, Iris, and Wine data sets). When categorical attributes are involved (e.g., Cancer, Credit, and Hepatitis data sets), the difference in the classification performances of the three different techniques is small.

The results also show that, for data sets with more than two classes (see Iris and Wine data sets), DA and polynomial classifiers perform very poorly compared to the minimum-distance classifier. For the Iris data set, DA and polynomial classifiers exhibit about the same classification performance. The difference in performance from the minimum-distance method is about 46% and can be explained by the fact that this data set has three classes. Likewise, for the Wine data set, DA and polynomial methods perform poorly compared to the minimum-distance method. However, in this particular case, the polynomial classifier is only 33.1% accurate, while the DA

TABLE V
CONFUSION MATRIX OF SAIS USING DIFFERENT CLASSIFICATION TECHNIQUES (EXCLUDING MISSING ATTRIBUTES)

Dataset	Minimum Distance			Discriminant Analysis			Polynomial		
Cancer		**0**	**1**		**0**	**1**		**0**	**1**
	0	40	2	**0**	44	0	**0**	44	0
	1	1	25	**1**	2	22	**1**	2	22
Credit		**0**	**1**		**0**	**1**		**0**	**1**
	0	33	4	**0**	33	6	**0**	24	6
	1	3	26	**1**	2	25	**1**	13	22
Diabetes		**0**	**1**		**0**	**1**		**0**	**1**
	0	49	10	**0**	49	10	**0**	51	10
	1	1	17	**1**	4	14	**1**	4	12
Hepatitis		**0**	**1**		**0**	**1**		**0**	**1**
	0	5	1	**0**	6	0	**0**	7	0
	1	0	2	**1**	1	1	**1**	1	0
Ionosphere		**0**	**1**		**0**	**1**		**0**	**1**
	0	24	2	**0**	25	3	**0**	24	3
	1	0	9	**1**	0	7	**1**	1	7
Iris		**0**	**1**	**2**	**0**	**1**	**2**	**0**	**1**	**2**
	0	4	0	0	**0**	0	0	0	**0**	0	0
	1	0	5	0	**1**	3	8	1	**1**	4	6	3
	2	0	0	6	**2**	0	0	3	**2**	0	0	2
Wine		**0**	**1**	**2**	**0**	**1**	**2**	**0**	**1**	**2**
	0	4	0	0	**0**	8	0	1	**0**	0	0	0
	1	0	4	0	**1**	0	4	5	**1**	3	8	7
	2	0	0	10	**2**	0	0	0	**2**	0	0	0

classifier is 55.5% accurate. The reason that the polynomial classifier performs even worse than the DA classifier is the fact that the Wine data set contains 13 attributes, and as such, the polynomial equation will have powers of up to 13.

To be able to determine whether SAIS has a linear computational complexity, the same experiment was performed using only half the size of each data set. Table VI shows the CPU times in millisecond of each experiment done on the whole data sets as well as on half the size of the data sets. The results prove that SAIS has a linear computational complexity as indicated by the ratio values (average of 0.5). The ratio value indicates that, if half the size of the data set is used for the experiment, the CPU time will be halved. In addition, the CPU times show that the polynomial technique is the most computationally intensive, followed by minimum distance and DA techniques.

B. Experiment 2—Including Missing Attributes

Disregarding observations with missing attributes is an option that can be used if the relative amount of missing data is small [41]. This is the case for the Breast Cancer and Credit Approval data sets, which have 2.3% and 5.4% of observations with at least one missing attribute, respectively. However, the Hepatitis data set contains 48.4% of observations with at least one missing attribute, and as such, the approach of excluding these observations is not appropriate since many good observations (even though they have missing attributes) are being lost. In addition, based on Table II, only 80 observations were used for the Hepatitis data set, and this amount is certainly not enough to train and test SAIS properly.

The more training data available, the more scenarios the model will be exposed to and the better will be the training process, assuming that the quality of data is good. The classification performance of SAIS on these three data sets can be further improved by including those observations with missing attributes. In order to do that, some minor changes were made to the model for handling missing attributes.

As stated in Section IV-C1, the exemplar-based method is based on the minimum distance from a class to classify each antigen. The smaller the distance between an exemplar and an antigen, the more likely the antigen will be classified in the class of that particular exemplar. Since all the data were normalized and their values are between zero and one, missing attributes are handled by returning a distance of one, which is the maximum possible distance to any attribute. The new distance function between an exemplar and an antigen is thus defined as

$$dist(x_{1,i}, x_{2,i}) = \begin{cases} 1, & \text{if missing} \\ catDist(x_{1,i}, x_{2,i}), & \text{if categorical} \\ contDist(x_{1,i}, x_{2,i}), & \text{if continuous.} \end{cases} \quad (7)$$

DA and polynomial classification methods are function-based classifiers [refer to (5) and (6)]. As such, it would not make any sense to replace the missing attributes by a number and allocate that number to x since this will directly affect the classification score (y) The two function-based classifiers were therefore not used in the second experiment.

Three data sets with missing attributes were partitioned into a training and a testing set, which is similar to what was done in the first experiment. The same training and testing procedures were performed on the modified SAIS, using only the minimum-distance classification method, and the results (classification performance and standard deviation) are shown in Table VII. Only the Cancer data set recorded an increase in classification performance (94.6%) when instances with missing attributes were included in the training and testing sets.

The difference in the classification performance of SAIS when applied to the Cancer and Credit data sets with and without missing attributes is not significant (0.2% for Cancer and 0.1% for Credit). This is to be expected since these two data sets have a relatively small number of observations with missing attributes. However, while a better performance for the Hepatitis data set was expected as there are more instances for training and testing the model, the classification performance decreased by 3%. The cause of this decrease was investigated by examining the instances with missing attributes, and it was found that 11.7% of the attributes of these observations were missing. It was also found that this particular data set did not have a good balance of instances from both classes, with the ratio of one class to another being 32 : 123. Thus, the quality of the data was not very good, and this probably explained why including those observations with missing attributes resulted in a decrease in classification performance.

C. Results

The experiments that were performed show that minimum distance is by far the most consistent method compared to

TABLE VI
COMPARISON OF CPU TIME OF SAIS USING DIFFERENT CLASSIFICATION TECHNIQUES ON WHOLE AND HALF DATA SETS

Classifier	Dataset size	Cancer	Credit	Diabetes	Hepatitis	Ionosphere	Iris	Wine
	Whole	34,156	55,674	32,997	9,067	30,978	5,806	7,348
Minimum Distance	Half	17,825	28,760	16,452	5,083	16,847	3,392	4,174
		[0.5]	[0.5]	[0.5]	[0.6]	[0.5]	[0.6]	[0.6]
	Whole	9,347	14,669	20,369	2,797	15,066	3,231	3,514
Discriminant Analysis	Half	5,084	8,052	10,967	1,931	8,866	2,000	2,209
		[0.5]	[0.5]	[0.5]	[0.7]	[0.6]	[0.6]	[0.6]
	Whole	68,677	166,813	182,042	21,406	320,650	13,384	37,337
Polynomial	Half	35,111	84,236	92,133	11,473	161,764	6,938	19,445
		[0.5]	[0.5]	[0.5]	[0.5]	[0.5]	[0.5]	[0.5]

[ ] - Ratio

TABLE VII
COMPARISON OF PERFORMANCE OF SAIS ON MISSING ATTRIBUTES DATA SETS (MINIMUM-DISTANCE METHOD)

Missing Attributes	Cancer		Credit		Hepatitis	
	n		n		n	
Excluding	683	94.4%	653	**86.2%**	80	**87.5%**
		(2.5)		(2.6)		(10.2)
Including	699	**94.6%**	690	86.1%	155	84.5%
		(2.3)		(3.9)		(7.5)

( ) - Standard deviation

DA and polynomial techniques, which perform very poorly on data sets with more than two classes (see Table IV). However, when only two classes are involved, minimum distance and DA maintain about the same classification accuracy. Polynomial, on the other hand, depends heavily on the number of attributes present in the data set since the number of powers in its equation is dependent on the number of attributes. As the latter increases, the complexity of the equation increases, and its performance decreases.

Table VIII shows the classification performance of SAIS when compared to some other classifiers obtained from [28] and [40]. It should be noted that, while three different types of classification methods were applied to SAIS, only the best classification performance among the three methods on each data set (refer to Tables IV and VII for best classification performance) is shown in Table VIII. It should also be noted that the classification performance of other classifiers for the Credit data set has not been included in Table VIII because they were not available. Moreover, there were no results for CLONALG, SSAIS, and RLAIS simply because these classifiers did not make use of these data sets. The full name of the classifiers shown in Table VIII can be found in Appendix.

It was found that, on average, SAIS classifies well for data sets with few attributes (< 10) and classes (< 4). The difference in performance between the best classifier of each data set and SAIS is small, except for the Ionosphere data set, which contains a large number of attributes (34 attributes), where the difference is 11.2%. SAIS even outperforms AIRS in the Diabetes and Iris data sets. It was also found that the classification accuracy of SAIS tends to decrease as the number of attributes and classes increase. There is clearly a need to improve this particular aspect of the classifier.

The results obtained also show that the performance of the SAIS model is in accordance to the *no free lunch* theory, which states that the performance of a system over one class of problems is offset by the performance over another class of problem [42]. Clearly, the performance of the model is quite competitive for data sets with small number of attributes; however, it is not that good for those with large number of attributes.

VI. CONCLUSION AND FUTURE WORK

In this paper, a novel and SAIS algorithm and classifier was implemented to remove some of the drawbacks (population control and local and one-shot optimizations) of other AIS classifiers without sacrificing the classification performance of the system. SAIS was tested on seven benchmark data sets using different classification techniques, and it was found to be a very competitive classifier.

Future work lies in improving the classification performance of SAIS for data sets with a large number of attributes (> 10) and classes (> 4). Multiple exemplars per class will be used to improve classification performance. An average and a k-nearest neighbor distance method will also be used, instead of the minimum-distance method. Replacing the HEOM with a heterogeneous value difference metric (HVDM) distance function is also envisaged since HVDM tends to produce better results than HEOM [36]. This is to be used on data sets with a mixture of a large number of categorical and continuous attributes, and a large number of classes.

APPENDIX
NAME OF CLASSIFIERS

The full name of the abbreviations used for the different classifiers, against which SAIS was compared, is given as follows:

AIRS	artificial immune recognition system;
ASI	assistant I tree;
ASR	assistant relief tree;
BP	backpropagation;
CART	classification and regression tree;
DA	discriminant analysis;

TABLE VIII
COMPARISON OF SAIS AND OTHER CLASSIFIERS CLASSIFICATION PERFORMANCE

Rank	Cancer		Diabetes		Hepatitis		Ionosphere		Iris		Wine	
	Name	**%**	**Name**	**%**	**Name**	**%**	**Name**	**%**	**Name**	**%**	**Name**	**%**
1	NB + kernel est	97.5	Logdisc	77.7	Weighted 9-NN	92.9	3-NN	98.7	Gronian	100.0	IncNet	98.9
2	SVM	97.2	IncNet	77.6	18-NN	90.2	VSS	96.7	SSV	98.0	SSV, opt-prune	98.3
3	Naive MFT	97.1	DIPOL92	77.6	FSM with rotations	89.7	IB3	96.7	C-MLP2LN	98.0	kNN	97.8
4	kNN	97.1	LDA	77.5	15-NN	89.0	1-NN	96.0	PVM 2 rules	98.0	SSV, opt-node	97.2
5	GM	97.0	**SAIS**	**77.4**	FSM without rotations	88.5	MLP+BP	96.0	**SAIS**	**97.3**	**SAIS**	**97.1**
6	VSS	96.9	SMART	76.8	**SAIS**	**87.5**	AIRS	94.9	PVM 1 rule	97.3	FSM	96.1
7	FSM	96.9	GTO DT	76.8	VSS	86.5	C4.5	94.9	AIRS	96.7		
8	Fisher LDA	96.8	kNN	76.7	LDA	86.4	RIAC	94.6	FuNe-I	96.7		
9	MLP + BP	96.7	ASI	76.6	Naive Bayes + Semi-NB	86.3	C4	94.0	NEFCLASS	96.7		
10	**SAIS**	**96.6**	Fisher DA	76.5	IncNet	86.0	SVM	93.2	CART	96.0		
11	LVQ	96.6	MLP + BP	76.4	QDA	85.8	FSM	92.8	FuNN	95.7		
12	SNB	96.6	LVQ	75.8	1-NN	85.3	Non-linear perceptron	92.0				
13	IB1	96.3	LFC	75.8	ASR	85.0	DB CART	91.3				
14	...	...	...	...	...	...	...	...				
18	RBF	95.9	BP	75.2	ASI	82.0	**SAIS**	**87.5**				
19	GTO DT	95.7	SSV	75.0	LFC	81.9	GTO DT	86.0				
20	ASI	95.6	CART DT	74.7	RBF	79.0						
21	OCN2	95.2	DB CART	74.4								
22	IB3	95.0	ASR	74.3								
23	MML tree	94.8	AIRS	74.1								
24	ASR	94.7	C4.5	73.0								

FSM feature space mapping;
FuNN fuzzy neural network;
GM Gaussian mixture;
GTO DT global tree optimization decision tree;
IB iterative Bayes;
IncNet incremental network;
kNN k nearest neighbor;
LDA linear discriminant analysis;
LFC lookahead feature construction;
LogDisc logistic discriminant;
LVQ learning vector quantization;
MFT multiple frequency tracker;
MLP multilayer perceptron;
MML minimum message length;
NB naive Bayes;
NEFCLASS neuro-fuzzy classification;
NN neural networks;
OCN2 overlap compensation neurons 2;
PVM parallel virtual machine;
QDA quadratic discriminant analysis;
RBF radial basis function;
RIAC rule induction algorithm based on approximate classification;
SAIS simple artificial immune system;
SNB seminaive Bayesian;
SSV separability of split value;
SVM support vector machine;
VSS variable shape search.

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their useful comments, which helped in improving the quality of this paper.

REFERENCES

[1] G. P. Zhang, "Neural networks for classification: A survey," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 30, no. 4, pp. 451–462, Nov. 2000.
[2] D. J. Hand and W. E. Henley, "Statistical classification methods in consumer credit scoring: A review," *J. R. Stat. Soc.*, vol. 160, no. 3, pp. 523–541, 1997.
[3] V. Srinivasan and Y. H. Kim, "Credit granting: A comparative analysis of classification procedures," *J. Finance*, vol. 42, no. 3, pp. 665–681, Jul. 1987.
[4] V. S. Desai, D. G. Convay, J. N. Crook, and G. A. Overstreet, "Credit scoring models in the credit union environment using neural networks and genetic algorithms," *IMA J. Math. Appl. Bus. Ind.*, vol. 8, no. 4, pp. 323–346, 1997.
[5] R. Malhotra and D. K. Malhotra, "Evaluating consumer loans using neural networks," *Int. J. Manag. Sci.*, vol. 31, no. 2, pp. 83–96, 2003.
[6] A. Tarakanov and D. Dasgupta, "A formal model of an artificial immune system," *BioSystems*, vol. 55, no. 1, pp. 155–158, Feb. 2000.
[7] J. Timmis, M. Neal, and J. Hunt, "An artificial immune system for data analysis," *BioSystems*, vol. 55, no. 1–3, pp. 143–150, Feb. 2000.
[8] M. Neal, "An artificial immune system for continuous analysis of time-varying data," in *Proc. 1st ICARIS*, Canterbury, U. K., 2002, pp. 76–85.

[9] O. Engin and A. Doyen, "A new approach to solve hybrid shop scheduling problems by an artificial immune system," *Future Gener. Comput. Syst.*, vol. 20, no. 6, pp. 1083–1095, Aug. 2004.
[10] J. E. Hunt and D. E. Cooke, "An adaptive, distributed learning system based on the immune system," in *Proc. IEEE Int. Conf. Syst., Man, and Cybern.*, 1995, pp. 2494–2499.
[11] D. Dasgupta and S. Forrest, "Novelty detection in time series data using ideas from immunology," in *Proc. ICSA 5th Int. Conf. Intell. Syst.*, Reno, NV, Jun. 1996, pp. 87–92.
[12] A. Watkins, "AIRS: A resource limited artificial immune classifier," M.S. thesis, Mississippi State Univ., Mississippi State, MS, Dec. 2001.
[13] D. W. Bradley and A. M. Tyrrell, "Immunotronics: Hardware fault tolerance inspired by the immune system," in *Proc. 3rd ICES*, Edinburgh, U. K., Apr. 2000, vol. 1801, pp. 11–20.
[14] D. Dasgupta, "Immunity-based intrusion detection system: A general framework," in *Proc. 22nd NISSC*, 1999, pp. 147–160.
[15] O. Nasraoui, D. Dasgupta, and F. González, "A novel artificial immune system approach to robust data mining," in *Proc. GECCO*, New York, Jul. 2002, pp. 356–363.
[16] C. A. Janeway, P. Travers, M. Walport, and J. D. Capra, *Immunobiology: The Immune System in Health and Disease*, 4th ed. Amsterdam, The Netherlands: Elsevier, 1999.
[17] J. E. Hunt and D. E. Cooke, "Learning using an artificial immune system," *J. Netw. Comput. Appl.*, vol. 19, no. 2, pp. 189–212, 1996.
[18] N. K. Jerne, "Towards a network theory of the immune system," *Ann. Immunol. (Inst. Past.)*, vol. 125C, no. 1/2, pp. 373–389, 1974.
[19] A. S. Perelson, "Immune network theory," *Immunol. Rev.*, vol. 110, pp. 5–36, 1989.
[20] F. M. Burnet, "Clonal selection and after," in *Theoretical Immunology*. New York: Marcel Dekker, 1978, pp. 63–85.
[21] L. N. de Castro and J. Timmis, "Artificial immune systems as a novel soft computing paradigm," *Soft Comput.*, vol. 7, no. 8, pp. 526–544, 2003.
[22] L. N. de Castro and F. J. Von Zuben, "The clonal selection algorithm with engineering applications," in *Proc. GECCO*, Las Vegas, NV, Jul. 2000, pp. 36–37.
[23] L. N. de Castro and F. J. V. Zuben, "Learning and optimization using clonal selection principle," *IEEE Trans. Evol. Comput.*, vol. 6, no. 3, pp. 239–251, 2001.
[24] J. A. White and S. M. Garrett, "Improved pattern recognition with artificial clonal selection?" in *Proc. ICARIS*. Berlin, Germany: Springer-Verlag, 2003, vol. 2787, pp. 181–193.
[25] J. Timmis and T.Knight, "Artificial immune systems: Using the immune system as inspiration for data mining," in *Data Mining: A Heuristic Approach*. Hershey, PA: Idea Group Publishing, 2002, pp. 209–230.
[26] J. D. Farmer, N. H. Packard, and A. S. Perelson, "The immune system, adaptation and machine learning," *Phys. D*, vol. 2, no. 1–3, pp. 187–204, Oct./Nov. 1986.
[27] J. Timmis and M. Neal, "A resource limited artificial immune system for data analysis," *Knowl.-Based Syst.*, vol. 14, no. 3, pp. 121–130, Jun. 2001.
[28] A. Watkins, J. Timmis, and L. Boggess, "Artificial immune recognition system (AIRS): An immune-inspired supervised learning algorithm," *Genet. Program. Evolvable Mach.*, vol. 5, no. 3, pp. 291–317, Sep. 2004.
[29] A. Watkins, "Exploiting immunological metaphors in the development of serial, parallel, and distributed learning algorithms," Ph.D. dissertation, Univ. Kent, Kent, U. K., Mar. 2005.
[30] M. Neal, J. Hunt, and J. Timmis, "Augmenting an artificial immune network," in *Proc. IEEE Syst., Man, and Cybern.*, 1998, vol. 4, pp. 3821–3826.
[31] H. Zheng, J. Zhang, and S. Nahavandi, "Learning to detect texture objects by artificial immune approaches," *Future Gener. Comput. Syst.*, vol. 20, no. 7, pp. 1197–1208, Oct. 2004.
[32] Y. Zhong, L. Zhang, B. Huang, and P. Li, "An unsupervised artificial immune classifier for multi/hyperspectral remote sensing imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 44, no. 2, pp. 420–431, Feb. 2006.
[33] E. Hart and P. Ross, "Exploiting the analogy between the immune system and sparse distributed memories: A system for clustering non-stationary data," *Genet. Program. Evolvable Mach.*, vol. 4, no. 4, pp. 333–358, 2003.
[34] H. S. Wilf, *Algorithms and Complexity*, 2nd ed. Wellesley, MA: A.K. Peters Ltd., 2002.
[35] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Mach. Learn.*, vol. 6, no. 1, pp. 37–66, Jan. 1991.
[36] D. R. Wilson and T. R. Martinez, "Improved heterogeneous distance functions," *J. Artif. Intell. Res.*, vol. 6, no. 1, pp. 1–34, 1997.
[37] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugen.*, vol. 7, no. 1, pp. 179–188, 1936.
[38] C. L. Blake and C. J. Merz, *UCI Repository of Machine Learning Databases*, 1998. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html
[39] J. W. Grzymala-Busse and M. Hu, "A comparison of several approaches to missing attribute values in data mining," in *Proc. 2nd Int. Conf. Rough Sets and Current Trends Comput.,* W. Ziarko and Y. Yao, Eds., 2001, pp. 378–385.
[40] W. Duch, *Data Sets Used for Classification: Comparison of Results*, 2000. [Online]. Available: http://www.phys.uni.torun.pl/kmk/projects/datasets.html
[41] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, ser. Springer Series in Statistics. New York: Springer-Verlag, 2001.
[42] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

Kevin Leung received the Bachelor of Business in business information systems (with honours) from the RMIT University, Melbourne, Australia. He is currently working toward the Ph.D. degree in the School of Business Information Technology, RMIT University.

His research interests include data mining and credit risk.

France Cheong received the Bachelor of Business degree in computing, the Master of Education degree in teaching, and the Ph.D. degree in computer systems engineering from the Royal Melbourne Institute of Technology University, Melbourne, Australia, and the master's degree in computer science from La Trobe University, Melbourne.

He is currently a Senior Lecturer with the School of Business Information Technology, RMIT University, Melbourne, Australia. His research interests include complex systems modeling and simulation using a range of intelligent systems and other techniques.

Christopher Cheong received the Bachelor of Applied Science in computer science (with honours) in 2003 from RMIT University. He is currently working toward the Ph.D. degree at the same university.

He is currently also a Lecturer with the School of Business Information Technology, RMIT University, Melbourne, Australia. His research interests include artificial intelligence, intelligent agents, evolutionary computing, and software engineering.